# Fairness-Aware Process Mining

Mahnaz Sadat Qafari(✉) and Wil van der Aalst

Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Aachen, Germany
{m.s.qafari,wvdaalst}@pads.rwth-aachen.de

**Abstract.** Process mining is a multi-purpose tool enabling organizations to improve their processes. One of the primary purposes of process mining is finding the root causes of performance or compliance problems in processes. The usual way of doing so is by gathering data from the process event log and other sources and then applying some data mining and machine learning techniques. However, the results of applying such techniques are not always acceptable. In many situations, this approach is prone to making obvious or unfair diagnoses and applying them may result in conclusions that are unsurprising or even discriminating. In this paper, we present a solution to this problem by creating a fair classifier for such situations. The undesired effects are removed at the expense of reduction on the accuracy of the resulting classifier.

## 1 Introduction

*Motivation.* Academic and commercial process mining tools aim to find the root causes of performance or compliance problems in processes. Mainly, a classifier, say a decision tree, is created using the data gathered from the process and then the rule mining is done using that decision tree [7]. However, this approach may lead to diagnoses that are not valuable. In some cases, the main cause of the problem is already known and essentially cannot be altered. Also, due to the strong correlation of the known main cause and the problem, it may become impossible to see the other minor but probably more practically valuable causes of the problem. Consider the following two scenarios: (i) there is a bottleneck in the process and it is caused by the busiest employee, or (ii) there are deviations caused by the most experienced resources taking the most difficult cases. In these scenarios, it is likely that the busiest employees or the most experienced resources are declared the main reasons for the bottleneck or deviations in the process. This is not just unfair but also does not provide novel insights (just stating the obvious). Even if we remove the attribute conveying the employee or the resource, still rules that proxy these attributes would be revealed as the result of the traditional rule mining [10]. In these cases, it is essential to make inference about the less trivial root-causes of the problem in the process.

As another application, consider that for a given process we are interested in questions which are related to investigating the process while ignoring the effect of different values of a particular attribute. "Following the progress of career paths while eliminating gender differences" is one example of these sorts of situations where we need to remove the correlation between two attributes in the data.

*Discrimination-Aware Data Mining.* Each population can be partitioned into several subgroups according to the properties of its members, e.g., race, age, or academic degree. *Discrimination* means treating a subgroup of people, called *sensitive group*, in an unfair way merely because of being a member of that subgroup. There is a possibility that negligent usage of new advanced technologies, especially in the field of data mining and machine learning, inadvertently cause discrimination. To avoid these phenomena, detecting discrimination and designing fair predictors have been studied intensively.

*Demographic parity* indicates the portion of people in the sensitive subgroup who receive the desired result must be the same as the whole population. To maintain this criterion, in some approaches the training data is manipulated [3,4, 10]. In another approach, [10], the representation of the data is changed, and the fairness is maintained as a side-effect of fair representations. In [5], demographic parity in a decision tree is retained by taking into account the information gain of the sensitive attribute as well as the class attribute as the criteria used for splitting the internal nodes. In [5], the relabeling technique is also used to further decrease the discrimination in the resulting decision tree. Besides demographic parity, other notions of fairness have been formalized in the literature. We refer the interested readers to [1] for a review of various fairness criteria.

*Process Mining.* Process mining is the link between model-based process analysis and data-oriented analysis techniques; a set of techniques that support the analysis of business processes based on event logs. In this context, several works have been dedicated to decision mining and finding the correlation among the process data and making predictions [2,6,7].

Ethical and legal effects of process mining can be considered in two categories; confidentiality and fairness issues. Confidentiality in the process mining has recently received attention [9]. To the best of our knowledge, there is no work in the area of process mining dedicated to investigating fairness issues. This is the first publication considering discrimination within a given process. The extended version of this paper is available in [8].

*Our Results.* We provide a solution for the previously mentioned problems. Specifying a problem in the process, we propose an approach by adopting the techniques available in data mining for removing discrimination from classifiers in the area of process mining to avoid unfair or obvious conclusions in such scenarios. We do that by declaring the attribute that indicates the existence of the problem in the given situation as the *class attribute* and the attribute that we want to decrease its dependency to the class attribute as the *sensitive attribute.* We consider the class attribute to be binary with the following two values: + indicates the desirable result conveying the problem of interest has not been faced while − has the opposite meaning. The sensitive attribute is also assumed to be binary, where ☹ convey belonging to the sensitive group while ☺ convey belonging to the rest of the population (favorable group). Now, we can consider the problem as a discriminatory case and remove the dependency of the class and the sensitive attributes in the resulting classifier by creating a fair classifier. Doing so, the resulting rules would not be discriminatory against the sensitive group. Also, this technique masks some of the causes of that problem and focus on the other ones.

The rest of the paper is organized as follow. In Sect. 2, we present the problem statement. A high-level overview of the proposed approach is presented in Sect. 3. The experimental results of applying the implemented method on a real event log are presented in Sect. 4. Finally, in Sect. 5, we summarize our approach and discuss directions for further research.

## 2 Problem Statement

To analyze conformance and performance problems, we use event data and process models (discovered or hand-made).[1] An event log is a collection of traces and each trace is a collection of events related to the same case. Also each trace may be associated with some attributes. Consider $\mathcal{U}_{act}$ as the universe of all possible *activity names*, $\mathcal{U}_{time}$ the universe of all possible *timestamps*, $\mathcal{U}_{att}$ the universe of all possible *attribute names*, $\mathcal{U}_{val}$ the universe of all possible *values*, and, $\mathcal{U}_{map} : \mathcal{U}_{att} \not\mapsto \mathcal{U}_{val}$. Also, let $values : \mathcal{U}_{att} \mapsto \mathbb{P}(\mathcal{U}_{val})$ be the function that returns the set of all possible values for each attribute name. We define an event log as follows:

**Definition 1 (Event Log).** *An* event *is an element of* $\mathcal{U}_{act} \times \mathcal{U}_{time} \times \mathcal{U}_{map}$ *and the universe of all possible* events *is denoted by* $\mathcal{E}$. *A* log *is an element of* $\mathbb{P}(\mathcal{U}_{map} \times \mathbb{P}(\mathcal{E}))$ *and the universe of all possible logs is denoted by* $\mathcal{L}$. *We call each* $t \in L$, *where* $L \in \mathcal{L}$, *a* trace.

stamp. To work with event logs, we need the following helper functions:

- Given an event $e = (act, time, map) \in \mathcal{E}$, $\pi_{act}(e) = act$, $\pi_{time}(e) = time$, and, $\pi_{map}(e) = map$.
- Given $t = (map, E) \in L$, where $L \in \mathcal{L}$, then $\pi_{map}(t) = map$ and $\pi_{events}(t) = E$.
- Given $E \in \mathbb{P}(\mathcal{E})$, then $\pi_{maxtime}(E) = \arg\max_{e \in E} \pi_{time}(e)$, $\pi_{act}(E) = \{e \in E | \pi_{act}(e) = act\}$, and, $E_{\leq time} = \{e \in E | \pi_{time}(e) \leq time\}$.

We assume that each event in a given log $L$ has a unique timestamp.

If the problem in the process is about the traces, like delay in some cases, then for a given trace all the values of its trace and event-level attributes might be relevant. However, if the problem is related to a specific activity, like a bottleneck in activity *act*, then we need to extract the data from the trace attributes plus the attributes of a subset of its events that occur before the occurrence of that specific event. Also, the class attribute may occur several times in a given trace. We define the notion of a *situation* to handle such cases as follows:

[1] We assume the reader to be familiar with the concepts like set, multi-set, and function. Given a non-empty set $X$, we denote all the non-empty subsets of $X$ by $\mathbb{P}(X)$. Given two sets $A$ and $B$, a partial function $f : A \not\mapsto B$ is defined as a function $f : A' \mapsto B$ for some $A' \subseteq A$. We say $f(a) = \perp$ if $a \notin A'$.

**Definition 2 (Situation).** *We define a* situation *as an element in* $(\mathcal{U}_{map} \times \mathbb{P}(\mathcal{E}))$. *The set of all possible situations is denoted by* $\mathcal{U}_{sit}$. *Given a log* $L \in \mathcal{L}$, *we define the set of all situations derived from it as:*

$$S_L = \bigcup_{(map,E) \in L} \Big( \bigcup_{e \in E} \{(map, E_{\leq \pi_{time}(e)})\} \Big).$$

*It is obvious that* $L \subseteq S_L$. *Any* $S \subseteq S_L$ *is called a* situation subset *of* $L$. *For a given log* $L$, *there are two main types of situation subsets. The first one is the* trace situation subset *which is* $S_{L,\perp} = L$. *The second type is the* event specified situation subsets *which includes all* $S_{L,act} = \{(map, E) \in S_L | \pi_{act}(\pi_{maxtime}(E)) = act\}$, *where* $act \in \mathcal{U}_{act}$ *and* $S_{L,act} \neq \emptyset$.

To specify an attribute, besides the name of the attribute, we need to know if it is a trace or an event attribute and if it is an event attribute, we need to know to which events does it belong. To concretely specify an attribute, we use the *situation feature* notion defined as follows:

**Definition 3 (Situation Feature).** *For any given* $a \in \mathcal{U}_{act} \cup \{\perp\}$ *and* $att \in \mathcal{U}_{att}$, *we call* $sf_{a,att} : \mathcal{U}_{sit} \not\rightarrow \mathcal{U}_{val}$ *a* situation feature. *Given a situation* $(map, E)$, *we define* $sf_{a,att}((map, E))$ *as follows:*

$$sf_{a,att}((map, E)) = \begin{cases} map(att) & a = \perp \\ \pi_{map}(\pi_{maxtime}(\pi_a(E)))(att) & a \in \mathcal{U}_{act} \end{cases}.$$

*We denote the universe of all possible situation features by* $\mathcal{U}_{sf}$. *Given a situation feature* $sf_{a,att}$, *we define* $values(sf_{a,att}) = values(att)$. *Also, for a given* $n \in \mathbb{N}$, $EP \in \mathcal{U}_{sf}^n$ *is a* situation feature extraction plan *of size* $n$, *where* $\mathcal{U}_{sf}^n$ *is defined as* $\underbrace{\mathcal{U}_{sf} \times \cdots \times \mathcal{U}_{sf}}_{n \ times}$.

A situation feature extraction plan can be interpreted as the schema, the tuple composed of those situation features that are relevant to the given problem in the process.

The first step of solving any problem is concretely specifying the problem. We call such a problem description a *situation specification* which is defined as follows:

**Definition 4 (Situation Specification).** *A situation specification is a tuple* $SS = (EP, ssf, csf, \epsilon)$ *in which*

*(i)* $EP \in \mathcal{U}_{sf}^n$, *where* $n \in \mathbb{N}$, *is the situation feature extraction plan which includes all the situation features for which we are going to investigate their effect on the given problem.*
*(ii)* $ssf \in \mathcal{U}_{sf}$, *the sensitive situation feature where* $values(ssf) = \{☺, ☹\}$ *and* $ssf \notin EP$.
*(iii)* $csf \in \mathcal{U}_{sf}$, *the class situation feature where* $values(csf) = \{+, -\}$, $csf \notin EP$, *and* $csf \neq ssf$.

*(vi)* $\epsilon \in [0, 1]$*, indicating the acceptable level of discrimination against* $ssf$ *(the amount of acceptable dependency between* $ssf$ *and* $csf$*).*

For a given situation specification, we go through the following three steps: (1) Enriching the log, (2) Extracting the data, and (3) Learning fair classifier. The general approach of our method is depicted in Fig. 1.

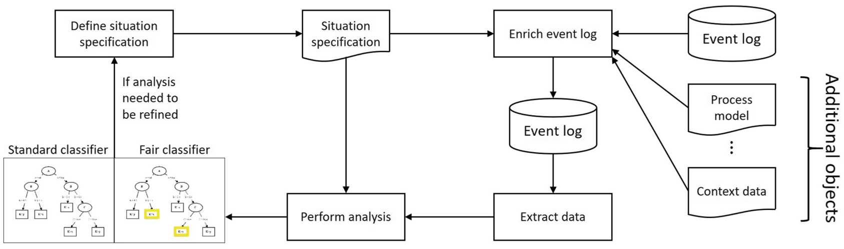

**Fig. 1.** The general framework proposed for fair root-cause analysis. First, according to the situation specification the event log is enriched by preprocessing the log and other sources of information. Then, the data is extracted from the enriched event log. Finally, two standard and fair classifier are created. Based on the analysis result, it is possible to adapt the situation specification to gather additional insights.

## 3 Approach

We go through every one of the steps for creating a fair classifier for a given situation specification, mentioned in the previous section, in more details.

*1. Enriching the log.* Let $SS = (EP, ssf, csf, \epsilon)$ be the given situation specification. If $EP$ includes situation features that can not be directly extracted from the given log $L$, we enrich the log by augmenting each trace of it. In this step, we add some attribute values to the traces and its events. These added attributes can be related to any one of different process characteristics; time perspective, data flow-perspective, control-flow perspective, conformance perspective, or resource organization perspective. They may be driven from the given log, conformance checking results from replaying the traces on a given Petri-net model, or any external information resource like the weather information.

*2. Extracting the data.* To discover meaningful dependency results by the classifier, we need to capture the data such that the causality relations among them and the class attribute are not violated. To do so, given $csf = sf_{a,att}$, we apply the following two rules while extracting the data:

1. If $a \in \mathcal{U}_{act}$, each trace may map to several situations and the data should be extracted from that part of the trace that happens before the occurrence of $csf$. However, if $a = \perp$, then $csf$ is related to a trace level attribute and the data should be extracted from the whole trace.
2. The value of the independent situation feature with the closest occurrence time to the occurrence of $csf$ must be collected.

The second rule is valid assuming that if one of the independent situation features has happened several times before the occurrence of *csf* in a trace, the one that is chronologically closest to the occurrence of *csf*, has the most effect on it.

To follow the first rule, for the given log $L$ and the situation specification $SS = (EP, ssf, csf, \epsilon)$, where $csf = sf_{a,att}$, we set $S = S_{L,\perp}$ if $a = \perp$ and we set $S = S_{L,act}$ if $a = act$.

The final step for extracting the data is creating a data table and annotating each row of the table by adding the values of sensitive and class situation feature to it.

**Definition 5 (Situation Feature Table).** *Given a situation feature extraction plan* $EP = (sf_{a_1,att_1}, \dots, sf_{a_n,att_n})$*, and a situation set* $S \subseteq \mathcal{U}_{sit}$*, a* situation feature table *is a multi-set which is defined as:*

$$T_{S,EP} = [(sf_{a_1,att_1}(s), \dots, sf_{a_n,att_n}(s)) | s \in S].$$

*For a log* $L \in \mathcal{L}$*,* $S \subseteq S_L$*, we call* $T_{S,EP}$ *a* situation feature table of $L$*.*

*For a given situation feature table* $T_{S,EP}$ *and* $csf, ssf \in \mathcal{U}_{ssf}$ *for which* $ssf \neq csf$ *and* $csf, ssf \notin EP$ *and* $\forall_{s \in S}\ (csf(s) \neq \perp \wedge ssf(s) \neq \perp)$*, we define an* annotated situation table $AT_{S,EP,ssf,csf}$ *as:*

$$AT_{S,EP,ssf,csf} = [(sf_{a_1,att_1}(s), \dots, sf_{a_n,att_n}(s), ssf(s), csf(s)) | s \in S].$$

*We call each element of* $AT_{S,EP,ssf,csf}$ *an* instance*. For a given instance* $inst_s = (sf_{a_1,att_1}(s), \dots, sf_{a_n,att_n}(s), ssf(s), csf(s))$ *we define* $\pi_{EP}(inst_s) = \big(sf_{a_1,att_1}(s), \dots, sf_{a_n,att_n}(s)\big)$*,* $\pi_{ssf}(inst_s) = ssf(s)$*, and,* $\pi_{csf}(inst_s) = csf(s)$*.*

Here, *ssf* is the sensitive and *csf* is the class (label) situation feature. Also, each member of $inst_s \in AT_{S,EP,ssf,csf}$ where $s \in S$ can be seen as a row of the data table in which $\pi_{EP}(inst_s) = (sf_{a_1,att_1}(s), \dots, sf_{a_n,att_n}(s))$ is the tuple including independent attribute values and $\pi_{csf}(inst_s) = csf(inst_s)$ is the class attribute value of $inst_s$.

*3. Learning fair classifier.* We define a classifier as follows:

**Definition 6 (Classifier).** *Let* $S$ *be a set of situations and* $EP = (sf_{a_1,att_1}, \dots, sf_{a_n,att_n})$ *be a situation extraction plan and* $csf \in \mathcal{U}_{sf}$ *such that* $\forall_{1 \leq i \leq n} sf_{a_i,att_i} \neq csf$*, then a* classifier *is a function* $class : T_{S,EP} \mapsto values(csf)$*.*

Given a classifier *class* and an annotated situation table $AT_{L,EP,ssf,csf}$, then the accuracy of *class* over $AT_{L,EP,ssf,csf}$ is measured as:

$$acc(class, AT_{L,EP,ssf,csf}) = \frac{|[inst \in AT_{L,EP,ssf,csf} | class(\pi_{EP}(inst)) = \pi_{csf}(inst)]|}{|AT_{L,EP,ssf,csf}|}.$$

For fairness, we use demographic parity as the main concept. To measure the discrimination in the data, we use the measure mentioned in [5], which is:

$$disc(AT_{L,EP,ssf,csf}) = \frac{|[inst \in AT_{L,EP,ssf,csf} | \pi_{ssf}(inst) = ☺ \wedge \pi_{csf}(inst) = +]|}{|[inst \in AT_{L,EP,ssf,csf} | \pi_{ssf}(inst) = ☺]|}$$

$$- \frac{|[inst \in AT_{L,EP,ssf,csf} | \pi_{ssf}(inst) = \text{☹} \wedge \pi_{csf}(inst) = +)]|}{|[inst \in AT_{L,EP,ssf,csf} | \pi_{ssf}(inst) = \text{☹}]|}.$$

By replacing $\pi_{csf}(inst)$ with $class(\pi_{EP}(inst))$ in this equation, we can measure the discrimination imposed by the classifier *class*.

For the classifier, we use decision trees. It is worth mentioning that both the classifier and the measure of discrimination can be changed according to the given application.

The first step toward removing the discrimination has already been taken during the creation of the classifier by not considering the sensitive situation feature for the classification purpose (Definition 6). As mentioned in many works, e.g. [10], this is not enough due to the existence of correlation among different situation feature values in a given situation feature table. The discrimination in the classifier can be further eliminated by relabeling technique. In this paper, we relabel leaves in the decision tree to balance accuracy and fairness. However, other methods of creating a discrimination free classifiers can be used.

In the implemented plug-in two classifiers are generated. The first one is a tree classifier that is generated by J48 tree classifier implementation of C4.5 algorithm in WEKA package. Then, if the level of discrimination in the resulting decision tree is more than an acceptable threshold $\epsilon$, the leaves of the decision tree are relabeled to create a fair classifier. For the relabeling, we use an algorithm similar to the one mentioned in [5]. In [5], the leaves of the tree are ordered in descending order of the ratio of the discrimination gain and accuracy lose of relabeling each leaf. Then according to this order, leaves are relabeled until the discrimination in the classifier tree is lower than $\epsilon$. As mentioned in [5], the problem of finding the set of leaves to be relabeled such that the discrimination in the decision tree is lower than a given threshold $\epsilon$ with the lowest possible negative effect on the accuracy of the decision tree is equivalent to the knapsack problem. In the relabeling algorithm implemented in the ProM plug-in, we use dynamic programming and rounding to choose approximately the best possible set of leaves to be relabeled.

Note that in the context of process mining and root cause analysis, changing the class label from + to − and from − to + at the same time may not be desirable. So in some cases we may need to restrict the relabeling technique to just desirable or just undesirable labeled leaves of the tree. If we restrict the relabeling, there might be cases where the discrimination of the fair tree is higher than given $\epsilon$.

## 4 Implementation and Experimental Results

The approach presented in Sect. 3 has been implemented as a plug-in of ProM which is an open-source framework for process mining. The implemented plug-in is available under the name *Discrimination aware decision tree*. The inputs of the plug-in are the event log, the Petri-net model of the process, and, the conformance checking results of replaying the given log on the given model. The current implementation focuses on three types of problems in a given process: (1) routing problems, (2) deviation problems, and (3) performance problems.

To illustrate the fair analysis of these problems we use one real data log, the *hospital billing*[2] event log. We use the last 20000 traces of hospital billing log in the experiments which include 71188 activities. In this initial evaluation, we created a controlled experiment with a known ground truth. In each experiment, the discrimination is added to the event log artificially and then the altered log is used to evaluate the method and investigate the effect of removing discrimination on the accuracy of the created fair decision tree. In all the experiments the same setting has been used. For example in all the experiments $\epsilon = 0.05$ and there was no limit for applying relabeling technique. Also for each event log, the same set of independent situation features has been chosen and all the parameters for creating the decision tree were the same. 60% of the data has been used for training, and 40% of the data has been used for testing the classifier. The results of our experiment are depicted in Fig. 2. The charts respectively show the results of applying our technique when there is (a) a performance problem, (b) a routing problem, and (c) a conformance problem in the hospital billing process.

As is depicted in Fig. 2, we can reduce the discrimination on the sensitive group at the expense of some reduction at the accuracy of the classifier. As expected, as the level of discrimination increases in the data, the amount of the accuracy of the classifier that needs to be sacrificed for removing the discrimination increases. We need to be careful using this technique, as there are occasions where discrimination may be put on the favorable group. This phenomenon is also unfair. Surprisingly, in some cases like Fig. 2(c), the fair decision tree outperforms the standard decision tree in terms of accuracy. This phenomenon has been reported in [5] as well.

The chart in Fig. 3, demonstrates the level of discrimination in the fair decision tree and its accuracy for different settings of parameter $\epsilon$. As expected, the accuracy of the fair decision tree is lower when $\epsilon$ is smaller. Here, we use delay in traces as the class situation feature.

## 5 Conclusion

The first step toward enhancing a process by removing one of its performance or compliance problems is diagnosing the root causes of that problem. By using standard data mining techniques for detecting the causes, the results might be obvious and mainly regarding those parts of the process that can not be altered. To reveal other less vivid causes of the problem we need to mask the obvious ones. We did so by looking at the cause that we need to ignore its effect on the problem as the sensitive attribute. Then we remove the dependency between the sensitive and the class attributes from the created classifier. This is done at the expense of a small reduction in the accuracy of the resulting classifier.

This research has several applications; detecting the discrimination within a process, removing the discrimination from the process by replacing the fair

[2] https://data.4tu.nl/repository/collection:event_logs_real.

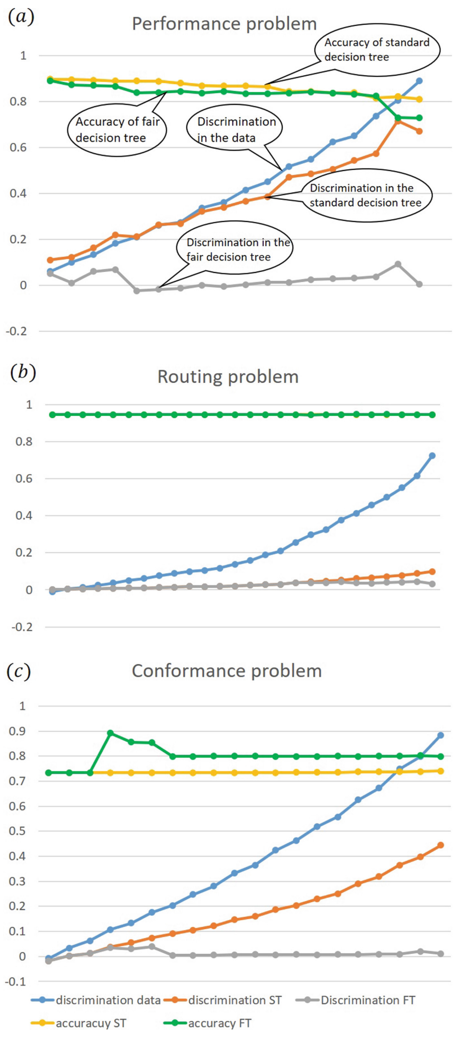

**Fig. 2.** The result of applying the implemented ProM plug-in on a real event log. In all these charts, the blue curve exhibits the level of discrimination in data (also by the $x$-axis.), the orange curve shows the level of discrimination in standard decision tree, the gray curve shows the level of discrimination in a fair decision tree, the yellow curve exhibits the accuracy of the standard decision tree, and, the green curve exhibits the accuracy of the fair tree. Chart (a) shows the results of applying our technique when there is a performance problem in the process for which we consider the delay in the traces. Chart (b) shows the results of applying our technique when there is a routing problem in the process for which we consider the choice between taking "BILLED" and skipping this transition in the hospital billing process. Chart (c) shows the results of applying our technique when there is a conformance problem in the process for which we consider the existence of deviation in the traces. In all these experiments $\epsilon = 0.05$. In all these experiments, the level of discrimination in the fair classifiers are less than the given threshold $\epsilon$. Also, as the level of discrimination increases in the data, the difference between the accuracy of the fair decision tree and standard decision tree increases. In chart (c), the fair decision tree demonstrates a better performance than the standard decision tree in terms of accuracy. (Color figure online)

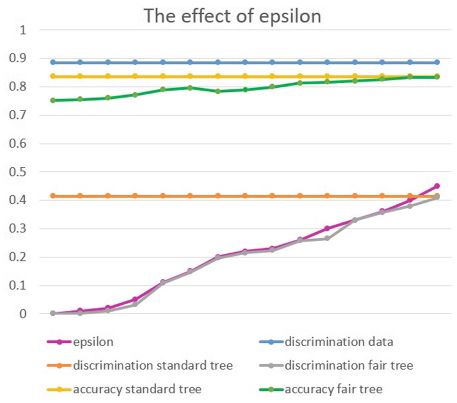

**Fig. 3.** The result of applying implemented plug-in with different values for parameter $\epsilon$ which is depicted in purple in the chart. In this chart, the value of $\epsilon$ shown by the pink curve. The level of discrimination in the data in all these experiments are the same. In all these experiments, the level of discrimination in the fair decision tree is lower than the given threshold $\epsilon$. Also, the accuracy of the fair decision tree tends to be lower for the lower values of $\epsilon$.

classifier with the current one, making more accurate and realistic judgments about the root causes of the problem at hand.

This research can be extended in several directions. The first one is to add new derived attributes to the log when enriching the log. The other one is altering the fairness criteria, the classification method, or the technique for creating the discrimination-free classifier depending on the application.

## References

1. Berk, R., Heidari, H., Jabbari, S., Kearns, M., Roth, A.: Fairness in criminal justice risk assessments: the state of the art. Sociol. Methods Res. 0049124118782533 (2018)
2. Fani Sani, M., van der Aalst, W., Bolt, A., García-Algarra, J.: Subgroup discovery in process mining. In: Abramowicz, W. (ed.) BIS 2017. LNBIP, vol. 288, pp. 237–252. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59336-4_17
3. Kamiran, F., Calders, T.: Classification with no discrimination by preferential sampling. In: Informal Proceedings of the 19th Annual Machine Learning Conference of Belgium and The Netherlands, Benelearn 2010, Leuven, Belgium, 27–28 May 2010, pp. 1–6 (2010)
4. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. Knowl. Inf. Syst. **33**(1), 1–33 (2012). https://doi.org/10.1007/s10115-011-0463-8
5. Kamiran, F., Calders, T., Pechenizkiy, M.: Discrimination aware decision tree learning. In: Proceedings of the 2010 IEEE International Conference on Data Mining, ICDM 2010, pp. 869–874. IEEE Computer Society, Washington, DC (2010). https://doi.org/10.1109/ICDM.2010.50

6. Leemans, S., Fahland, D., van der Aalst, W.: Process and Deviation Exploration with Inductive Visual Miner. pp. 46–50 (2014)
7. de Leoni, M., van der Aalst, W.M., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**(C), 235–257 (2016). https://doi.org/10.1016/j.is.2015.07.003
8. Qafari, M.S., van der Aalst, W.: arxiv: Fairness-aware process mining (2019). https://arxiv.org/abs/1908.11451
9. Rafiei, M., von Waldthausen, L., van der Aalst, W.M.P.: Ensuring confidentiality in process mining. In: Proceedings of the 8th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA 2018), Seville, Spain, 13–14 December 2018, pp. 3–17 (2018). http://ceur-ws.org/Vol-2270/paper1.pdf
10. Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C.: Learning fair representations. In: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML 2013, pp. III-325–III-333. JMLR.org (2013). http://dl.acm.org/citation.cfm?id=3042817.3042973